Week 12

- Streams
- Text and Binary Files
- Random Access Files
- Reading and Writing to Text Files
- Files Application – Extending Parts Program

Read Pages 283 - 289

Streams

- A stream is a device that allows data to flow between a computer program and I/O devices.
- We have used two streams which are pre-created
  - System.out output stream connected to screen
  - System.in input stream connected to keyboard

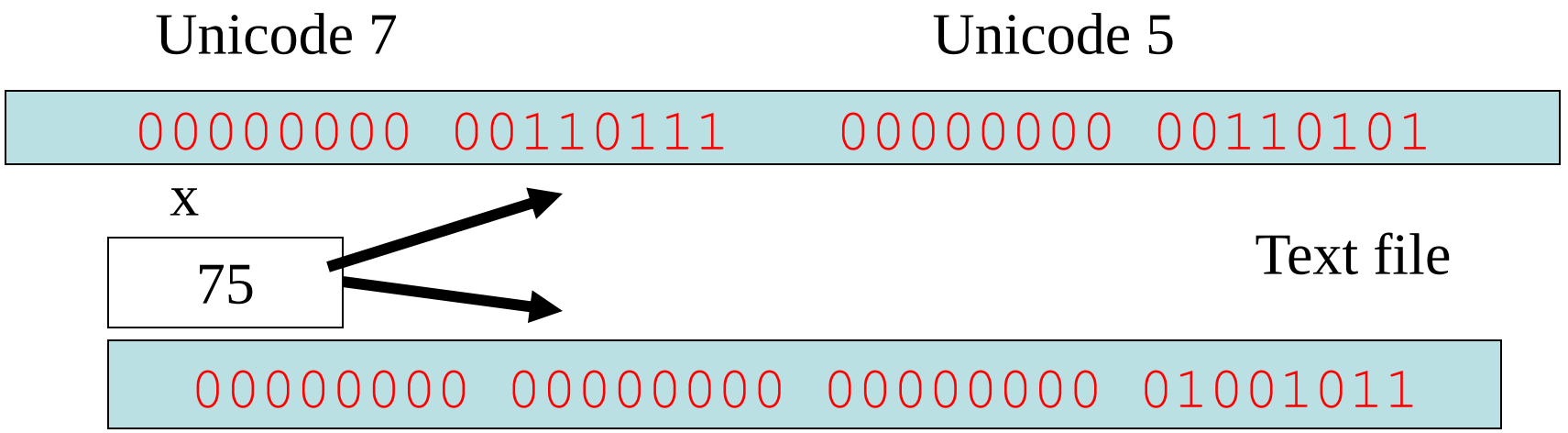
Files

- In the Parts program we developed we had to input all the part details through the keyboard every time! Any changes to parts are not reflected when we run the program again. Surely there must be a better way ?
- Write can write all the part object details to a file, allowing it to read back and recreated.
- In this course we will deal mainly with text files – though binary files are more efficient.
- Java also provides convenient ways to serialize (write) objects to files – but we will not be dealing with that topic in this course

```
p123 Axle 50 20 215.50
p124 Panel 120 30 115.50
p125 Brake-Pad 800 200 125.50
…
```

Binary and Text Files

- Contents of binary files are to be treated as binary digits.
  - Humans cannot make sense
  - Cannot be created with test editors
  - More efficient to process
- Contents of text file can be created by text editors and can be read read by humans.



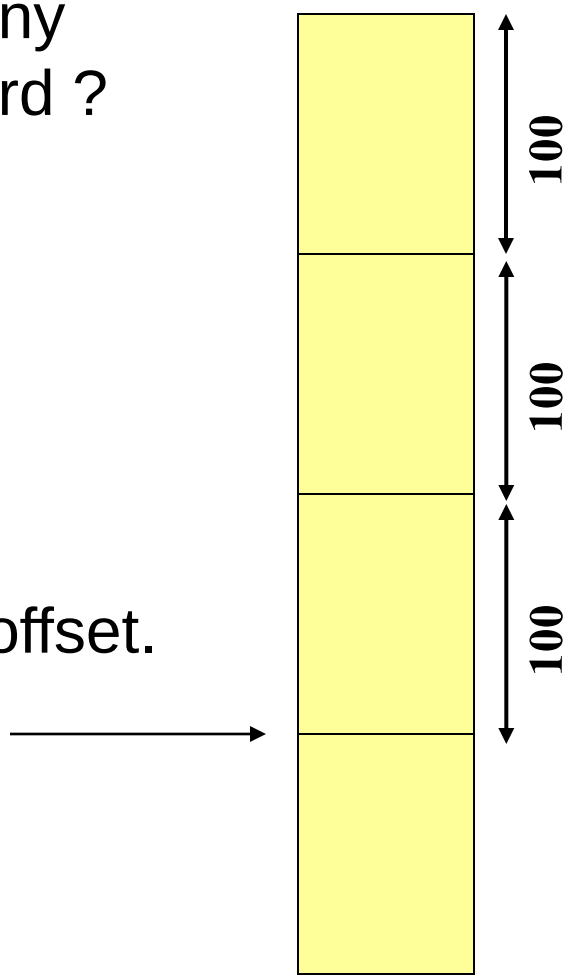
Random Access Files

If each record is 100 bytes how many bytes should I skip to read 4th record ?

To read nth record ?

Use the seek() method of RandomAccessFile specifying the offset.

Why is it useful ?



RandomAccessFile Example

```
import java.io.*;
public class RFile  {
   public static void main(String[] args) throws IOException  {
         RandomAccessFile rafile = new RandomAccessFile("test.dat","rw");
         // writing 3 int elements to array
         for (int i=1; i<=3; i++)
             rafile.writeInt(i);// each int occupies 4 bytes in the file

         //  reading and updating values
         rafile.seek( rafile.getFilePointer() - 4 * 1);
         int x=rafile.readInt();
         x = x + 20;
         rafile.seek( rafile.getFilePointer() - 4 * 1);
         rafile.writeInt(x);

         rafile.seek(0);
         for (int i=1; i<=3; i++)
             System.out.println(" " + rafile.readInt());
   }  // main
}
```

Reading a Text File - displaying it on the screen

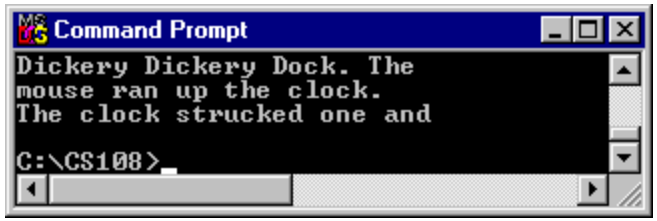
```
import java.io.*;
public class FileToScreen  {
   public static void main(String[] args)
      throws IOException  {

        // create buffered file stream
      BufferedReader bufr = new BufferedReader
                  (new FileReader ("source.txt"));
      String inputLine;

      inputLine = bufr.readLine();

      while (inputLine != null)  {  //  more input
          System.out.println(inputLine);
          inputLine = bufr.readLine();
      }
   }  // main
}
```

File name

FileReader instead of InputStreamReader

source.txt

```
Dickery Dickery Dock. The
mouse ran up the clock.
The clock strucked one and
```



Copying One File to Another

```
import java.io.*;
public class FileToFile2  {
   public static void main(String[] args)
      throws IOException  {

       // create buffered file stream
       BufferedReader bufr = new BufferedReader
                  (new FileReader ("source.txt"));
       // create new buffered output writer

       PrintWriter pw = new PrintWriter (new
          BufferedWriter(new FileWriter ("dest.txt")));


       String inputLine;

      inputLine = bufr.readLine();
      while (inputLine != null)  {  // still more input
         pw.println(inputLine);
         inputLine = bufr.readLine();
      }
      pw.close();  // flushes buffers & frees resources
   }  // main
}
```

Create a FileWriter object chain it to BufferedWriter and then to PrintWriter.

Using println() of PrintWriter.

Numbers and other types of data in text files

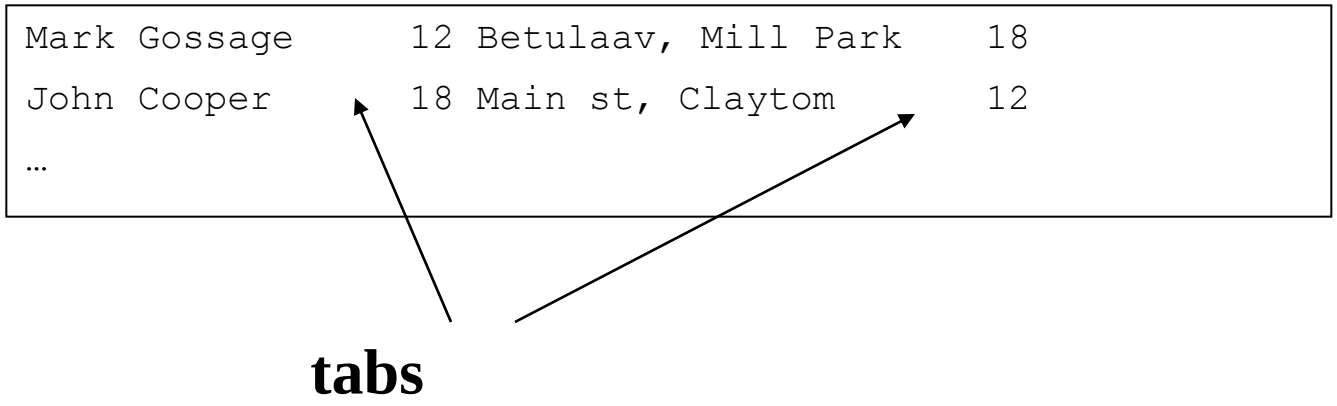
```
// writes numbers, booleans to a file as text
import java.io.*;
public class BinaryToTextFile  {
   public static void main(String[] args)
      throws IOException  {

     //  create new buffered output writer
     PrintWriter pw = new PrintWriter(new
       BufferedWriter(new FileWriter ("dest.txt")));

     boolean flag = true;
     int anInt = 17;
     double aDouble = 123.45;

     pw.println(flag);
     pw.println(anInt);
     pw.println(aDouble);

     pw.close();
   }  // main
}
```

Overloaded println() method

Writing a records with multiple fields

```
import java.io.*;
public class WriteStudentsInfo{
   public static void main(String[] args)
     throws IOException
    {
        String name, address, fileName;
        int age;
        System.out.println("Enter the name of the file : ");
        ConsoleReader console = new ConsoleReader(System.in);
        fileName = console.readLine();
        PrintWriter pw = new PrintWriter(new BufferedWriter
                                    (new FileWriter(fileName)));
        do {
            System.out.println("Enter name : "); name = console.readLine();
            System.out.println("Enter address:");address = console.readLine();
            System.out.println("Enter age : ");  age = console.readInt();
            pw.println(""+name+"\t"+address+"\t"+age);
            System.out.println("Continue  Y/N ?");
        } while ( console.readLine().charAt(0) == 'Y');
        pw.close();
   }
}
```

Format of the output file and Reading the values back



To reading these fields back we use a StringTokenizer class, whose constructor takes a String and a delimeter (“\t”)

```
StringTokenizer inReader =
      new StringTokenizer(line,"\t");
```

```
import java.io.*;import java.util.*;
public class ReadStudentsInfo
{  public static void main(String[] args)  throws IOException
   {  String line;    String name;
      String address;
      String fileName;   int age;
      System.out.println("Name of file to read from : ");
      ConsoleReader console = new ConsoleReader(System.in);
      fileName = console.readLine();
      BufferedReader br = new BufferedReader(new FileReader(fileName));
      System.out.println("name \t address \t age");
      while ( (line = br.readLine()) != null)
      {  StringTokenizer inReader = new StringTokenizer(line,"\t");
         if (inReader.countTokens() != 3)
            throw new IOException("Invalid Input Format");
         else
          {  name = inReader.nextToken();
            address = inReader.nextToken();
            age = Integer.parseInt(inReader.nextToken());
         }
         System.out.println(name+"\t"+address + "\t" + age);
      }
      br.close();
   }
}
```

Using Scanner class

- We used Scanner class for reading input from console.
- Scanner class can also be used for reading from a file
- To read from a file, you can chain a Scanner class to a File class as in:

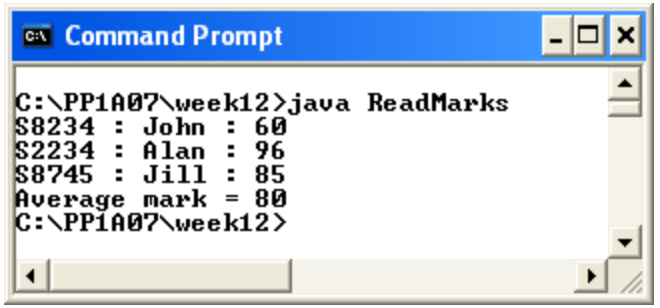
```
Scanner input = new Scanner(new File("marks.txt"));
```

- Scanner breaks the input into tokens based on whitespace characters.
- The next slide shows an example where a text file containing student marks is read and displayed in the screen. It also computes and displays the average mark.

Using Scanner class

```
import java.util.*;
import java.io.*;
public class ReadMarks
{
   public static void main(String[] args) throws Exception
   {
      Scanner input = new Scanner(new File("marks.txt"));
      // writing 3 int elements to array
      int sum = 0;
      int count = 0;
       while(input.hasNext())
      {
         String sNum = input.next();
         String firstName = input.next();
         int mark = input.nextInt();
          sum += mark;
          count++;
          System.out.println(sNum + " : " + firstName + " : " + mark);
       }
      if (count > 0)
          System.out.print("Average mark = " + sum/count);
      input.close();
   }  // main
}
```

S8234	John 60
S2234	Alan 96
S8745	Jill 85



Files : Application (Making objects persistent)

- Extend the Parts Management program so that all the object are written to a text file when user exit from the system
- Use the text file to recreate the objects when the system starts again.
- Choose an appropriate format for the text file
- You mau use the StringTokenizer or the split method of String to tokenize a line of text, when reading from a file.

Files : Application (Making objects Persistent)

Files : Application (Making objects Persistent)

Files : Application (Making objects Persistent)